UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,160	11/26/2003	Stephen H. Miller	2-4	9244

7590 03/03/2009

Ryan, Mason & Lewis, LLP
90 Forest Avenue
Locust Valley, NY 11560

EXAMINER	
SYED, FARHAN M	
ART UNIT	**PAPER NUMBER**
2165	

MAIL DATE	DELIVERY MODE
03/03/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.**	**Applicant(s)**	
	10/723,160	MILLER ET AL.	
	Examiner	**Art Unit**	
	FARHAN M. SYED	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 December 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7 and 10-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7 and 10-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-7 and 10-20 are pending. The Examiner acknowledges amended claims 1, 18, and 20 and cancelled claim 9.

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 29 December 2008 has been entered.

Response to Remarks/Argument

3. Applicant's arguments, see pages 8-9, filed 01 December 2008, with respect to the rejection(s) of claim(s) 1-7 and 10-20 under 35 U.S.C. 103(a) have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of Gai et al (U.S. 6,651,096 and known hereinafter as Gai).

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 1-7 and 10-16 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The Examiner notes that claim 1 recites "A method of generating a representation of an access control list, the representation being utilizable in a processor.." According to Applicant's disclosure, see page 6, lines 26-27, which state "The functionality of the network processor as described herein may be implemented at least in part in the form of **software program code,**" the processor does not inherently involve the use of a particular machine or apparatus and therefore the Examiner believes is software per se. See *In re Bilsk*i, 545 F3d 943, 88 USPQ2d 1385 (Fed. Cir. 2008)i.

The claims lack the necessary physical articles or objects to constitute a machine or a manufacture within the meaning of 35 USC 101. They are clearly not a series of steps or acts to be a process nor are they a combination of chemical compounds to be a composition of matter. As such, they fail to fall within a statutory category. They are, at best, functional descriptive material *per se*.

Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." Both types of "descriptive material" are nonstatutory when claimed as descriptive material *per se*, 33 F.3d at 1360, 31 USPQ2d at 1759. When functional descriptive material is recorded on some computer-readable medium, it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994)

Merely claiming nonfunctional descriptive material, i.e., abstract ideas, stored on a computer-readable medium, in a computer, or on an electromagnetic carrier signal, does not make it statutory. See *Diehr*, 450 U.S. at 185-86, 209 USPQ at 8 (noting that the claims for an algorithm in *Benson* were unpatentable as abstract ideas because "[t]he sole practical application of the algorithm was in connection with the programming of a general purpose computer.").

6. Claims 2-7 and 10-16 are dependent claims of claim 1 and do not cure the deficiencies in claim 1 and rejected for reasons similar.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-7 and 10-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admission of Prior Art (Pub. No. 2005/0114655 A1), hereinafter AAPA, in view of Cheriton (USPN 7,149,216), hereinafter Cheriton, and further in view of Buia et al. (USPN 2004/0078683 A1), hereinafter Buia and in further view of Gai et al (U.S. 6,651,096 and known hereinafter as Gai)(newly presented).

Per claim 1, *AAPA* discloses a method of generating a representation of an access control list **(See pg. 1 paragraph [0003] where routers or switches typically utilize ACLs.)**, the representation being utilizable in a processor **(See pg. 1 paragraph [0004] where network processors are used.)**, the method comprising the steps of:

determining a plurality of rules of the access control list, each of at least a subset of the rules having a plurality of fields and a corresponding action **(See page 1 paragraph [0003] where an ACL generally comprises a set of rules, the rules having fields and corresponding actions.).**

AAPA does not explicitly disclose processing the rules to generate a multi-level tree representation of the access control list, each of one or more of the levels of the tree representation being associated with a corresponding one of the fields; and wherein at least one level of the tree representation comprises a plurality of nodes.

However, *Cheriton* discloses the ACL having rules compiled into an ACL-M-trie Plus data structure having multiple levels, and each level having of a plurality of nodes being associated with fields, the fields included source and destination addresses **(See col. 2 lines 15-18 and 35-37, and col. 4 lines 5-9 where M-trie Plus data structure is a multi-level tree.)**. *Cheriton* also discloses wherein for each level of the tree representation that corresponds to a field of a rule of the access control list **(See *Cheriton* col. 4 lines 35-41 where first and second levels corresponding to fields including source and destination address.)**, a master list of nodes is maintained, each node comprising at least one of information characterizing one or more field values associated with that node **(See *Cheriton* col. 3 lines 53-67 where extended**

ACL List is master list.), one or more subtree pointers for that node, and a reference count indicating how many ancestor nodes are pointing to that node **(See *Cheriton* col. 3 lines 46-51 where oppointer includes pointers for a node and opcode; i.e. subtree pointers and a reference count.)**. *Cheriton* also discloses wherein the tree representation is generated by sequentially processing the rules of the access control list, the processing for a given rule comprising applying values of fields of the given rule to one or more existing nodes of the tree representation **(See col.1 lines 55-59 and col. 2 lines 15-19 of *Cheriton* for access control list processing.)**, and wherein when a particular value of a field of the given rule is applied to a given node **(See col. 2 lines 35-43 where sequence of nodes have applied source and destination address values, see col. 4 lines 5-9.)**.

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) (AAPA) to generate a multi-level tree representation of the access control list as taught by *Cheriton*. The motivation would have been to provide a faster way of traversing the ACL due to earlier methods being relatively slow **(See col. 1 lines 39-46 of Cheriton.)**.

AAPA in view of *Cheritan* does not explicitly disclose that with two or more of the nodes of a level having a common subtree, the tree representation including only a single copy of that subtree; the subtree comprising at least one node that is not a leaf node of the tree representation; the tree representation being characterizable as a directed graph in which each of the two nodes having the common subtree points to the single copy of the common subtree and a copy is made of the node, the field value is

applied to the copied node, and the resultant updated node is added to the master list of the corresponding level.

However, *Buia* discloses two or more of the nodes of a level of a tree in a directed graph representation having a common subtree pointing to a single copy of the common subtree comprising at least one node that is not a leaf node of the tree **(See Fig. 7B where two nodes 'FAULT A' and 'FAULT F' have common subtree at node 'FAULT C' where node 'FAULT C' of the common subtree is not a leaf node and the subtree is the only copy in the tree representation. The tree representation is characterized as a directed graph.)**. *Buia* discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level **(See pg. 8 paragraph [0099] Buia teaches creating copy of node.**).

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree representation (as *AAPA, Cheritan, and Buia)* to have two or more of the nodes of a level of the tree in a directed graph representation having a common subtree pointing to a single copy of the common subtree and discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level as taught by *Buia*. The motivation would have been to optimize efficiency and productivity by creating an ACL tree representation that handles identical tree portions or subtrees by sharing subtrees **(as seen on pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**.

AAPA in view of *Cheritan* and Buia does not explicitly wherein the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented.

Gai discloses wherein the updated node is compared with other nodes of the master list and if a duplicate node is found **(i.e. "Do the intersecting actions (i.e. nodes) conflict" The preceding text appears to compare other nodes of the master list to see if a duplicate node is found.)(Figure 9A)**, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node **(i.e. "Merge using conflict resolution table" The preceding text indicates that the step of merging includes the step of deleting the copied node and a pointer to the duplicate node is provided to an ancestor node that points to the given node)(See Figures 8, 9A, and 13 for illustration)**, a subtree pointer of the ancestor node is updated to the duplicate node pointer **(Figure 13 teaches a subtree pointer of the ancestor node is updated)(Figure 13)**, a reference count of the duplicate node now pointed to by the ancestor node is incremented **(i.e. "Set M=M+1" The preceding text clearly indicates that a reference count is incremented)(Figures 8 and 9A)** and a reference count of the given node previously pointed to by the ancestor node is decremented **(See Figure 7 that discloses a**

reference count is decremented by determining if the stack is empty. When the stack is empty, the loop of executing the processing for ACL is exited.)(Figure 7).

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree representation (as *AAPA, Cheritan, Buia, and Gai)* to have the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented **(as seen on pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**.

Per claim 2, *AAPA* discloses wherein the common subtree is implemented at least in part as a matching table **(AAPA See pg. 1 paragraph [0009] where ACL rules are stored in table format. Also see [0003] where ACL typically imply an ordered matching or ordered list of *AAPA*.)**.

Per claim 3, *Cheriton* discloses wherein the plurality of fields comprises at least first and second fields, the first field comprising a source address field and the second field comprising a destination address field **(See pg. 1 paragraph [0003] where fields**

define source and destination addresses of *Cheriton*.).

Per claim 4, *Cheriton* discloses wherein a final level of the tree representation comprises a plurality of leaf nodes, each associated with one of the actions of the plurality of rules **(See col. 2 lines 35-42, col. 3 lines 53-63, and col. 4 lines 5-9 of *Cheriton* where second level of nodes of the addresses is associated with routing rules.)**.

Per claim 5, *Cheriton* discloses wherein the at least one level of the tree representation comprises a root level of the tree representation **(See col. 4 lines 1-4 of *Cheriton* where tree, including roots; i.e. root level.)**.

Per claim 6, *Buia* wherein a second level of the tree representation includes a plurality of nodes, each being associated with a subtree of a given one of the plurality of nodes of the root level of the tree representation **(See Figs. 7B and Fig. 8 where tree representation may include plurality of root level nodes as in 7B and a second level with a plurality of nodes from a root level.*)***.

Per claim 7, *Cheriton* discloses wherein for each level of the tree representation that corresponds to a field of a rule of the access control list **(See *Cheriton* col. 4 lines 35-41 where first and second levels corresponding to fields including source and destination address.)**, a master list of nodes is maintained, each node comprising at

least one of information characterizing one or more field values associated with that node **(See *Cheriton* col. 3 lines 53-67 where extended ACL List is master list.)**, one or more subtree pointers for that node, and a reference count indicating how many ancestor nodes are pointing to that node **(See *Cheriton* col. 3 lines 46-51 where oppointer includes pointers for a node and opcode; i.e. subtree pointers and a reference count.)**.

Per claims 10, *Buia* discloses the method of claim 9 wherein if a duplicate node is found in the master list, that duplicate node is moved to an initial position in the master list **(See pg. 8 paragraph [0099] for copy node.)**.

Per claims 11, *Cheriton* discloses the wherein for each node in the master list **(See Cheriton were master list is extended ACL list)**, a copy pointer is maintained, and wherein when a copied node is compared to the master list and a duplicate node is found, the copied node is added as a copy to the master list for use in conjunction with the processing of a subsequent rule **(See AAPA for ACL rules. See Buia pg. 8 paragraph [0099] for copy node.)**.

Per claims 12, *Cheriton* and *Buia* discloses wherein for each node in the master list **(See Cheriton col. 3 lines 64-66 where extended ACL list is master list)**, a signature is maintained in order to facilitate node comparisons, a full comparison of node subtrees being performed only if a match is obtained between node signatures

(See Buia Fig. 7B for common subtree node.).

Per claims 13, *Cheriton* discloses wherein the signature for a given node is generated as a function of at least one of a field value and a subtree pointer **(See Cheriton col. 3 lines 46-51 for subtree pointer; i.e. oppointer and col. 4 lines 5-10 for field values; i.e. source and destination address.)**.

Per claim 14, *AAPA* in view of *Cheriton* and *Buia* discloses wherein the corresponding actions include at least an accept action and a deny action **(See rejection of claim 1 above where an accept or deny action is involved in routing the packets.)**.

Per claim 15, *AAPA* discloses the method of claim 1 further including the step of storing at least a portion of the tree representation in memory circuitry accessible to the processor **(See AAPA pg. 1 paragraph [0007] where memory is taught.)**.

Per claim 16, *AAPA* and *Cheriton* discloses the method of claim 1 further including the step of utilizing the stored tree representation to perform an access control list based function in the processor **(See AAPA pg. 1 paragraph [0004] for utilizing in the network processor, [0007] for memory, and *Cheriton* col. 2 lines 15-20 for stored tree structure.)**.

Per claim 17, *AAPA* discloses the method of claim 16 wherein the access control list based function comprises packet filtering **(See AAPA pg. 1 paragraph [0004] where packet filtering is taught)**.

Per claim 18, rejection of claim 1 is incorporated. Claim 18 is rejected under the same rationale as claim 1. *AAPA* in view of *Cheriton* and *Buia* discloses an apparatus configured for performing one or more processing operations utilizing a representation of an access control list, the access control list comprising a plurality of rules, each of at least a subset of the rules having a plurality of fields and a corresponding action **(See AAPA paragraph [0003] for ACL comprising rules having fields.)**, the apparatus comprising:

a processor having memory circuitry associated therewith **(See AAPA pg. 1 paragraph [0004] for network processors and [0007] for memory circuitry.)**;

the memory circuitry being configured for storing **(See AAPA pg. 1 [0007] for memory circuitry)** at least a portion of a multi-level tree representation of the access control list, each of one or more of the levels of the tree representation being associated with a corresponding one of the fields **(See Cheriton cols. 2 lines 35-44 for levels of multi-level tree representation of ACL.)**;

the processor being operative to utilize the stored tree representation to perform an access control list based function **(See AAPA pg. 1 paragraph [0004] for network processors in view of Cheriton cols. 2 lines 35-44 for tree representation to perform ACL function.)**

wherein at least one level of the tree representation comprises a plurality of nodes **(See col. 2 lines 15-18 and 35-37, and col. 4 lines 5-9 of Cheriton where M-trie Plus data structure is a multi-level tree.)**,

Cheriton also discloses wherein for each level of the tree representation that corresponds to a field of a rule of the access control list **(See *Cheriton* col. 4 lines 35-41 where first and second levels corresponding to fields including source and destination address.)**, a master list of nodes is maintained, each node comprising at least one of information characterizing one or more field values associated with that node **(See *Cheriton* col. 3 lines 53-67 where extended ACL List is master list.)**, one or more subtree pointers for that node, and a reference count indicating how many ancestor nodes are pointing to that node **(See *Cheriton* col. 3 lines 46-51 where oppointer includes pointers for a node and opcode; i.e. subtree pointers and a reference count.)**. *Cheriton* also discloses wherein the tree representation is generated by sequentially processing the rules of the access control list, the processing for a given rule comprising applying values of fields of the given rule to one or more existing nodes of the tree representation **(See col.1 lines 55-59 and col. 2 lines 15-19 of *Cheriton* for access control list processing.)**, and wherein when a particular value of a field of the given rule is applied to a given node **(See col. 2 lines 35-43 where sequence of nodes have applied source and destination address values, see col. 4 lines 5-9.)**.

AAPA in view of *Cheritan* does not explicitly disclose that with two or more of the nodes of a level having a common subtree, the tree representation including only a single copy of that subtree; the subtree comprising at least one node that is not a leaf

node of the tree representation; the tree representation being characterizable as a directed graph in which each of the two nodes having the common subtree points to the single copy of the common subtree and a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level.

However, Buia discloses two or more of the nodes of a level of a tree in a directed graph representation having a common subtree pointing to a single copy of the common subtree comprising at least one node that is not a leaf node of the tree **(See Fig. 7B where two nodes 'FAULT A' and 'FAULT F' have common subtree at node 'FAULT C' where node 'FAULT C' of the common subtree is not a leaf node and the subtree is the only copy in the tree representation. The tree representation is characterized as a directed graph.)**. *Buia* discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level **(See pg. 8 paragraph [0099] Buia teaches creating copy of node.***)*

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree representation (as *AAPA, Cheritan, and Buia)* to have two or more of the nodes of a level of the tree in a directed graph representation having a common subtree pointing to a single copy of the common subtree and discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level as taught by *Buia*. The motivation would have

been to optimize efficiency and productivity by creating an ACL tree representation that handles identical tree portions or subtrees by sharing subtrees **(as seen on pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**

AAPA in view of *Cheritan* and Buia does not explicitly wherein the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented.

Gai discloses wherein the updated node is compared with other nodes of the master list and if a duplicate node is found **(i.e. "Do the intersecting actions (i.e. nodes) conflict" The preceding text appears to compare other nodes of the master list to see if a duplicate node is found.)(Figure 9A)**, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node **(i.e. "Merge using conflict resolution table" The preceding text indicates that the step of merging includes the step of deleting the copied node and a pointer to the duplicate node is provided to an ancestor node that points to the given node)(See Figures 8, 9A, and 13 for illustration)**, a subtree pointer of the ancestor node is updated to the duplicate node pointer **(Figure 13 teaches a subtree pointer of the ancestor node is updated)(Figure 13)**, a reference count of the duplicate node now pointed to by the ancestor node is incremented **(i.e. "Set M=M+1"**

The preceding text clearly indicates that a reference count is incremented)(Figures 8 and 9A) and a reference count of the given node previously pointed to by the ancestor node is decremented **(See Figure 7 that discloses a reference count is decremented by determining if the stack is empty. When the stack is empty, the loop of executing the processing for ACL is exited.)(Figure 7).**

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree representation (as *AAPA, Cheritan, Buia, and Gai)* to have the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented **(as seen on pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**.

Per claim 19, rejection of claim 18 is incorporated. *AAPA* discloses the apparatus of claim 18 wherein the memory circuitry comprises at least one of internal memory and external memory of the processor **(See AAPA paragraph [0007] memory circuitry and [0004] for processor.)**

Per claim 20, rejection of claim 1 is incorporated. Claim 20 is rejected under the same rationale as claim 1. *AAPA* in view of *Cheriton* and *Buia* discloses an article of manufacture comprising a machine-readable storage medium having program code stored thereon, the program code generating a representation of an access control list, the representation being utilizable in a processor **(See AAPA pg. 1 paragraph [0003] for ACL [0004] for processor, and [0007] for article of manufacture comprising machine-readable storage medium, i.e. memory.)**, wherein the program code when executed implements the steps of:

determining a plurality of rules of the access control list, each of at least a subset of the rules having a plurality of fields and a corresponding action **(See AAPA page 1 paragraph [0003] where an ACL generally comprises a set of rules, the rules having fields and corresponding actions.)**; and

processing the rules to generate a multi-level tree representation of the access control list, each of one or more of the levels of the tree representation being associated with a corresponding one of the fields; wherein at least one level of the tree representation comprises a plurality of nodes **(See Cheritan where col. 2 lines 15-18 and 35-37, and col. 4 lines 5-9 where M-trie Plus data structure is a multi-level tree.)**.

Cheriton also discloses wherein for each level of the tree representation that corresponds to a field of a rule of the access control list **(See *Cheriton* col. 4 lines 35-41 where first and second levels corresponding to fields including source and destination address.)**, a master list of nodes is maintained, each node comprising at

least one of information characterizing one or more field values associated with that node **(See *Cheriton* col. 3 lines 53-67 where extended ACL List is master list.)**, one or more subtree pointers for that node, and a reference count indicating how many ancestor nodes are pointing to that node **(See *Cheriton* col. 3 lines 46-51 where oppointer includes pointers for a node and opcode; i.e. subtree pointers and a reference count.)**. *Cheriton* also discloses wherein the tree representation is generated by sequentially processing the rules of the access control list, the processing for a given rule comprising applying values of fields of the given rule to one or more existing nodes of the tree representation **(See col.1 lines 55-59 and col. 2 lines 15-19 of *Cheriton* for access control list processing.)**, and wherein when a particular value of a field of the given rule is applied to a given node **(See col. 2 lines 35-43 where sequence of nodes have applied source and destination address values, see col. 4 lines 5-9.)**.

AAPA in view of *Cheritan* does not explicitly disclose that with two or more of the nodes of a level having a common subtree, the tree representation including only a single copy of that subtree; the subtree comprising at least one node that is not a leaf node of the tree representation; the tree representation being characterizable as a directed graph in which each of the two nodes having the common subtree points to the single copy of the common subtree and a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level.

However, Buia discloses two or more of the nodes of a level of a tree in a directed graph representation having a common subtree pointing to a single copy of the

common subtree comprising at least one node that is not a leaf node of the tree **(See Fig. 7B where two nodes 'FAULT A' and 'FAULT F' have common subtree at node 'FAULT C' where node 'FAULT C' of the common subtree is not a leaf node and the subtree is the only copy in the tree representation. The tree representation is characterized as a directed graph.)**. *Buia* discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level **(See pg. 8 paragraph [0099] Buia teaches creating copy of node.*).***

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree representation (as *AAPA, Cheritan, and Buia)* to have two or more of the nodes of a level of the tree in a directed graph representation having a common subtree pointing to a single copy of the common subtree and discloses a copy is made of the node, the field value is applied to the copied node, and the resultant updated node is added to the master list of the corresponding level as taught by *Buia*. The motivation would have been to optimize efficiency and productivity by creating an ACL tree representation that handles identical tree portions or subtrees by sharing subtrees **(as seen on pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**

AAPA in view of *Cheritan* and Buia does not explicitly wherein the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to

the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented.

Gai discloses wherein the updated node is compared with other nodes of the master list and if a duplicate node is found **(i.e. "Do the intersecting actions (i.e. nodes) conflict" The preceding text appears to compare other nodes of the master list to see if a duplicate node is found.)(Figure 9A)**, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node **(i.e. "Merge using conflict resolution table" The preceding text indicates that the step of merging includes the step of deleting the copied node and a pointer to the duplicate node is provided to an ancestor node that points to the given node)(See Figures 8, 9A, and 13 for illustration)**, a subtree pointer of the ancestor node is updated to the duplicate node pointer **(Figure 13 teaches a subtree pointer of the ancestor node is updated)(Figure 13)**, a reference count of the duplicate node now pointed to by the ancestor node is incremented **(i.e. "Set M=M+1" The preceding text clearly indicates that a reference count is incremented)(Figures 8 and 9A)** and a reference count of the given node previously pointed to by the ancestor node is decremented **(See Figure 7 that discloses a reference count is decremented by determining if the stack is empty. When the stack is empty, the loop of executing the processing for ACL is exited.)(Figure 7).**

At the time of the invention, it would have been obvious to a person of ordinary skill in the art of generating Access Control Lists (ACLs) in a multi-level tree

representation (as *AAPA, Cheritan, Buia, and Gai)* to have the updated node is compared with other nodes of the master list and if a duplicate node is found, the copied node is deleted and a pointer to the duplicate node is provided to an ancestor node that points to the given node, a subtree pointer of the ancestor node is updated to the duplicate node pointer, a reference count of the duplicate node now pointed to by the ancestor node is incremented and a reference count of the given node previously pointed to by the ancestor node is decremented **(as seen *on* pg. 3 paragraph [0025] and pg. 9 paragraph [0101] of Buia.)**.

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Miller, Micheal J., et al (U.S. 6,996,662) and (U.S. Pub. 2003/0005146 A1) – teaches content addressable memory array having flexible priority support.

Cathey, Jim et al (U.S. 7,075,926) and (U.S. Pub. 2002/0085560 A1) – teaches The packet classification engine has a decision tree-based classification logic for classifying a packet. Each of the leaves of the tree represents a packet classification. The packet classification engine uses the header data cache index to retrieve the header data to perform multiple header checks, starting at a root of the tree and traversing branches until a leaf has been reached. The application engine has a number of programmable sub-engines arrayed in a pipelined architecture. The packet

classification engine provides start indicators based on the packet classification to the programmable sub-engines to identify application programs to be executed. The sub-engines includes a source lookup engine, a destination lookup engine and a disposition engine, which are used to make a disposition decision for the inbound packets in a processing pipeline.

Valois, Denis et al (U.S. Pub 2004/0260818) – teaches a system and method for providing compliance verification information in the field of network security, is disclosed. The software system, which is designed to be modular, provides compliance verification information with respect to a security policy. Each level of verification may be performed on a local basis, with respect to a network device, or on a global basis, with respect to multiple network devices, such as a communications network.

Contact Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Farhan M. Syed whose telephone number is 571-272-7191. The examiner can normally be reached on 8:30AM-5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Christian Chace can be reached on 571-272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Farhan M Syed/
Examiner, Art Unit 2165